



Sismio

Rapporto sull'implementazione

Federico Terzi 0000753482
Matteo Pellegrino 0000766387
Alberto Bagnacani 0000767392

Cos'è Sismio?



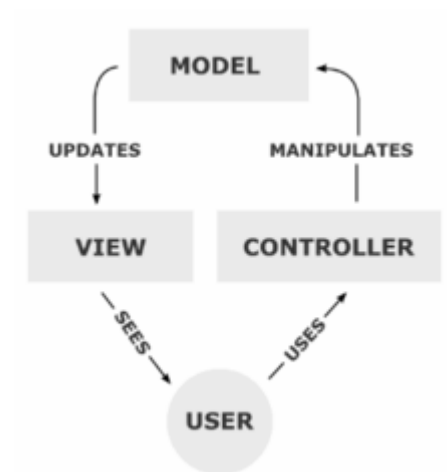
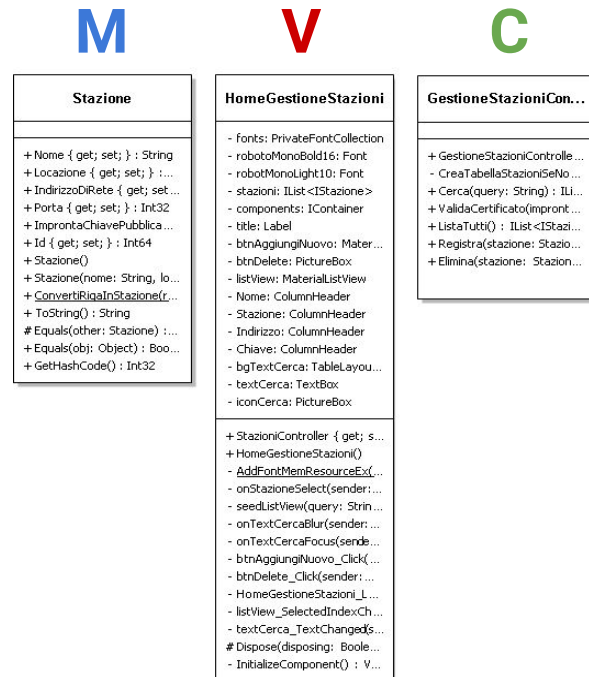
- Sistema **distribuito** per la rilevazione, l'analisi e la gestione di scosse sismiche nel territorio.
- Rende possibile la creazione di una **rete di stazioni** sismiche interconnesse tra loro.
- Scambio dati attraverso **connessioni sicure**.
- Dotato di un **sistema di permessi** per regolarne l'accesso.
- Fornisce uno **storico** di tutti gli eventi sismici passati.

MVC



Sismio realizzato tramite utilizzo di pattern **ModelViewController**.

- **Separazione** della logica di **presentazione** dei dati dalla logica di **business**
- **Maggiore rapidità** del processo di **sviluppo**



Singleton



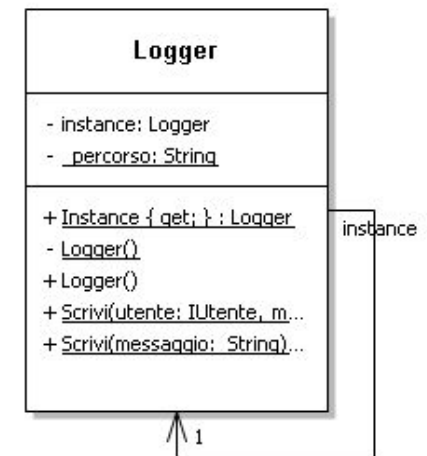
Logger realizzato tramite utilizzo di pattern **Singleton**.

- Logger possiede **una** ed una **sola istanza**
- Punto di **accesso globale** a tale istanza
- Istanziazione **“pigra”** e **statica**

```
// Logger.cs
public sealed class Logger
{
    private static readonly Logger instance = new Logger();

    public static Logger Instance
    {
        get
        {
            return instance;
        }
    }

    public static int Scrivi(string messaggio)
    {
        try
        {
            fs = new FileStream(_percorso, FileMode.Append);
            using (StreamWriter writer = new StreamWriter(fs))
            {
                // ...
            }
        }
    }
}
```



```
1528548835 Registrata stazione: Nome: ReggioEmiliaA1, Locazione: Reggio Emilia, Indirizz
1528548894 Registrata stazione: Nome: GubbioA1, Locazione: Gubbio, IndirizzoDiRete: 10.
1528549051 Registrata stazione: Nome: CatanzaroA1, Locazione: Catanzaro, IndirizzoDiRet
1528549137 Registrata stazione: Nome: MondolfoA1, Locazione: Mondolfo, IndirizzoDiRete:
1528549310 Eliminata stazione: Nome: ReggioEmiliaA1, Locazione: Reggio Emilia, Indirizz
1528549358 Registrato utente: Nome: Pluto, Cognome: Paperonis, Email: pluto@gmail.com,
1528549363 Eliminato utente: Nome: tizio, Cognome: caio, Email: tizio@caio.it, Username
```

Database



Persistenza basata su **SQLite**.

- Libreria software che implementa **DBMS SQL** di tipo **ACID**
- Compatto, veloce, open-source, nessuna dipendenza esterna

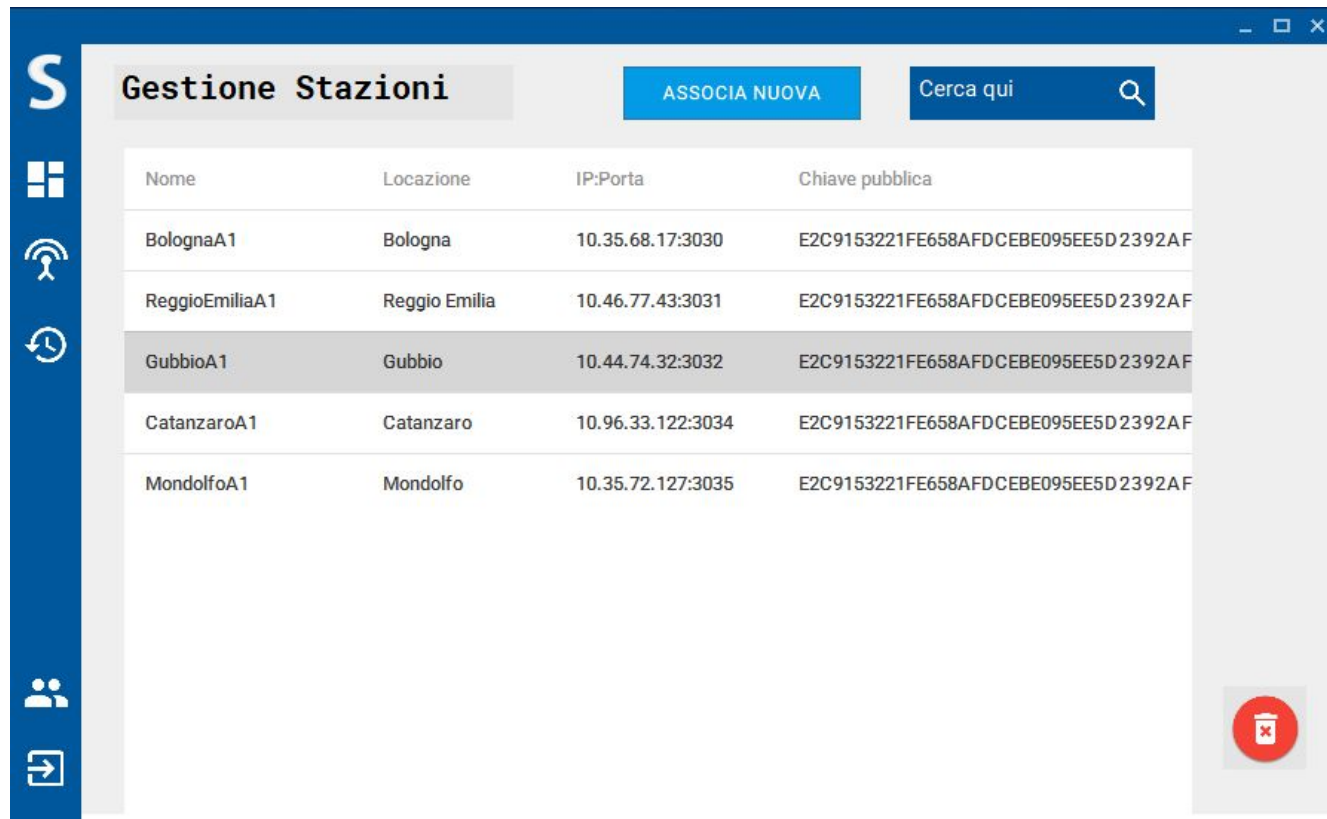
```
public bool Elimina(Stazione stazione)
{
    // ...
    using (SQLiteCommand cmd = new SQLiteCommand(_connection))
    {
        cmd.CommandText = "DELETE FROM stazioni WHERE id = @Id";
        cmd.Prepare();
        cmd.Parameters.AddWithValue("@Id", stazione.Id);
        try
        {
            risultato = cmd.ExecuteNonQuery();

            if (risultato > 0)
                Logger.Scrivi("Eliminata stazione: " + stazione);
        }
        catch (SQLiteException e)
        {
            Console.WriteLine(e.ToString());
        }
    }
}
```

Home: Gestione stazioni

Interfaccia grafica per la **gestione** delle **stazioni**.

- **Semplicità** di utilizzo ed **immediatezza** delle informazioni



Nome	Locazione	IP:Porta	Chiave pubblica
BolognaA1	Bologna	10.35.68.17:3030	E2C9153221FE658AFDCEBE095EE5D2392AF
ReggioEmiliaA1	Reggio Emilia	10.46.77.43:3031	E2C9153221FE658AFDCEBE095EE5D2392AF
GubbioA1	Gubbio	10.44.74.32:3032	E2C9153221FE658AFDCEBE095EE5D2392AF
CatanzaroA1	Catanzaro	10.96.33.122:3034	E2C9153221FE658AFDCEBE095EE5D2392AF
MondolfoA1	Mondolfo	10.35.72.127:3035	E2C9153221FE658AFDCEBE095EE5D2392AF

Form: Login



Rappresenta ViewAutenticazione.

Visualizzata all'avvio e al logout.

Responsabile di mostrare MainForm se l'accesso è valido.

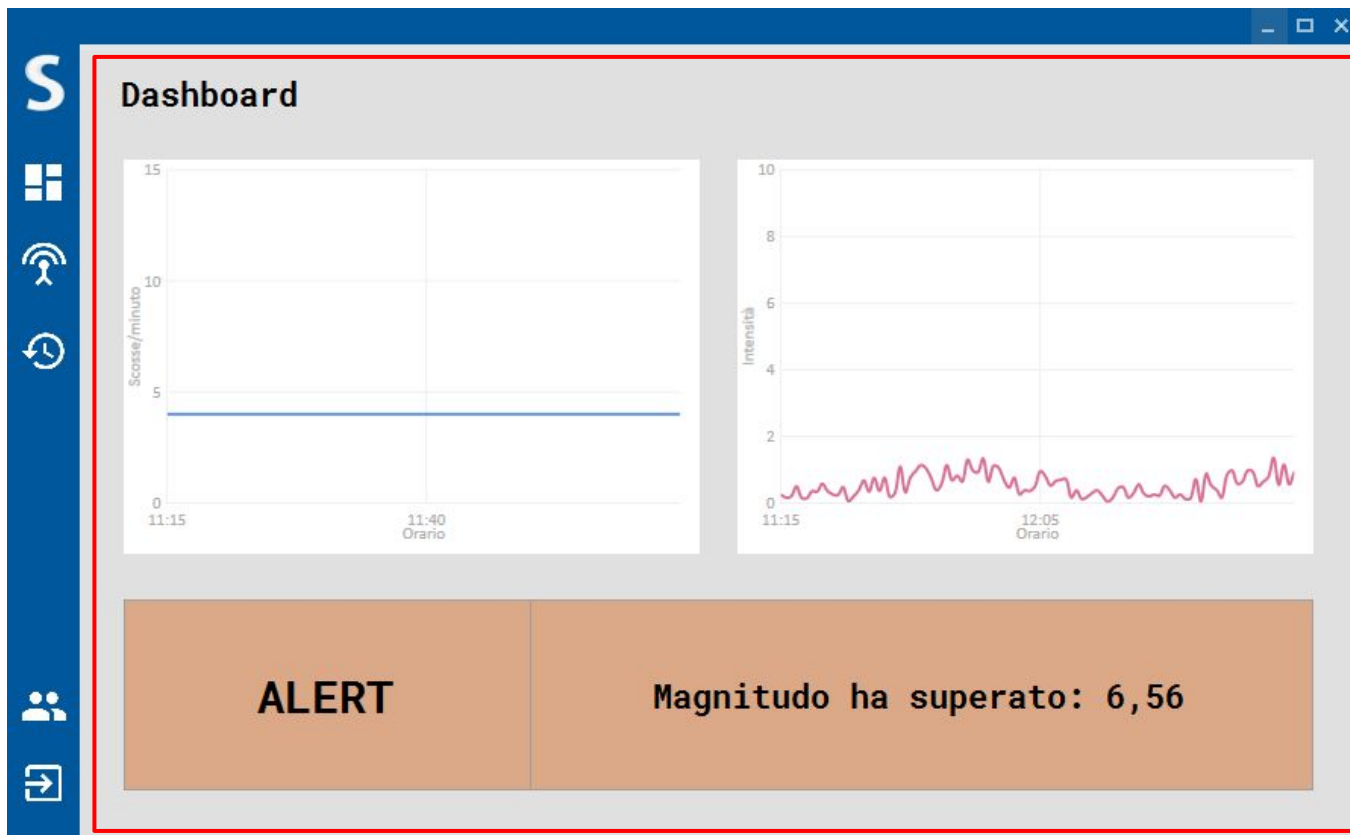
Login remoto e locale.

The screenshot shows a login window with a dark blue background. At the top center is the 'Sismio' logo in white. Below the logo is a login form with three input fields: a text field containing 'admin', a password field with six asterisks, and a dropdown menu labeled 'Locale'. Below these fields is a blue button with the text 'ACCEDI' in white. The window has standard OS window controls (minimize, maximize, close) in the top right corner.

Form: MainForm



Raccoglie tutti gli UserControl, uno per ogni maschera Home.
Dispone della barra di navigazione.
Limita la visibilità ai componenti se i permessi non sono sufficienti.



UserControl
HomeDashboard
Contenuto dinamico

UserControl



- **Suddivisione logica di business**
Una UserControl per ogni Home.
Le Home raccolgono le View correlate.
La logica di business richiede l'iniezione dei Controller ed è specifica ad ogni UserControl.
- **Devono risiedere in una Form**
Ogni UserControl Home è istanziata in MainForm.

Dependency Injection:

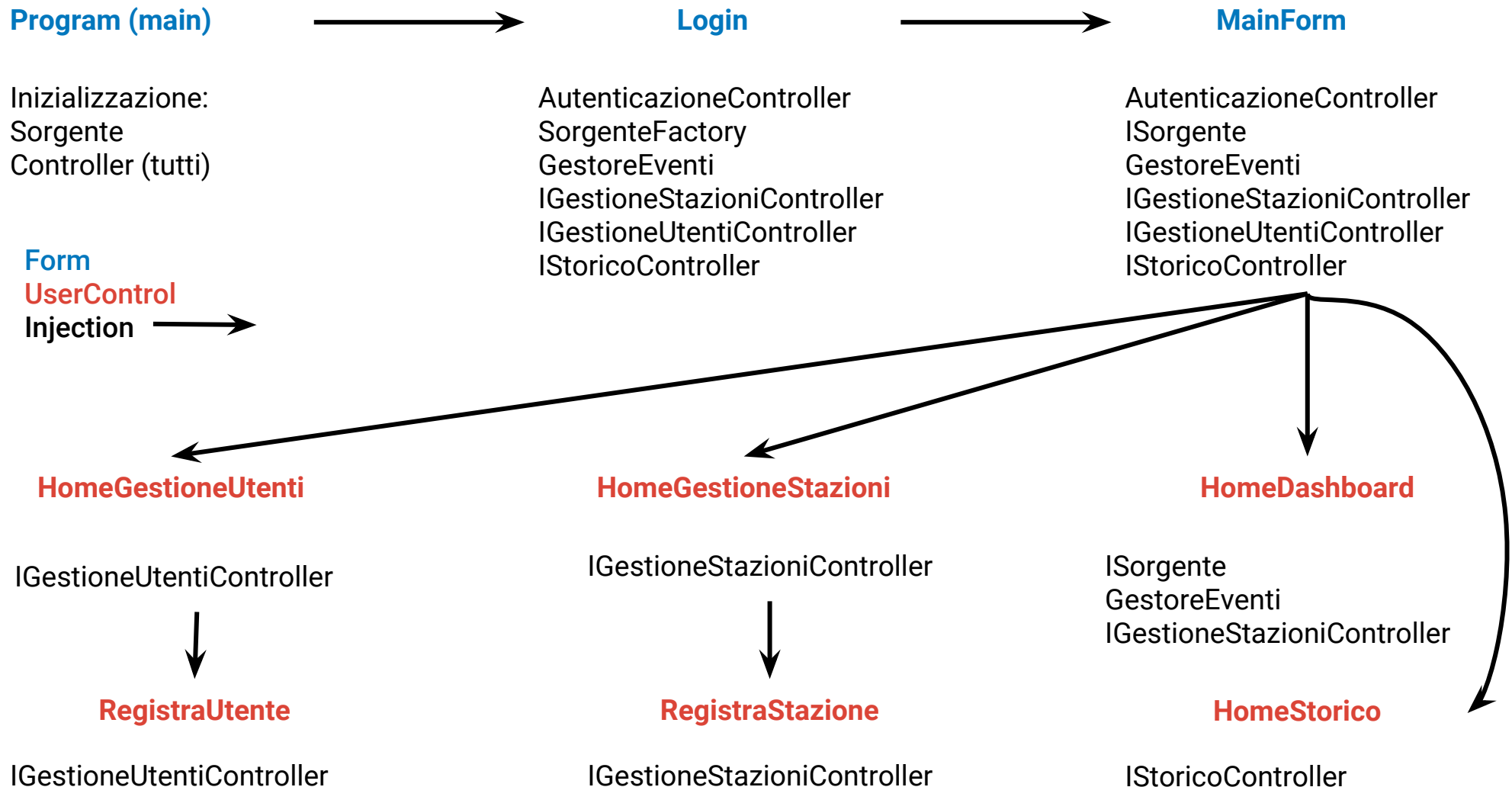
View



- Dipendenze dei componenti
Injection tramite setter di property.
Ogni Form e UserControl mantiene property dei Controller necessari alla logica di business correlata.
- Dependency Inversion Principle
Disaccoppiare i componenti.
- Minimizzazione dei Singleton necessari

Dependency Injection:

View



Librerie utilizzate



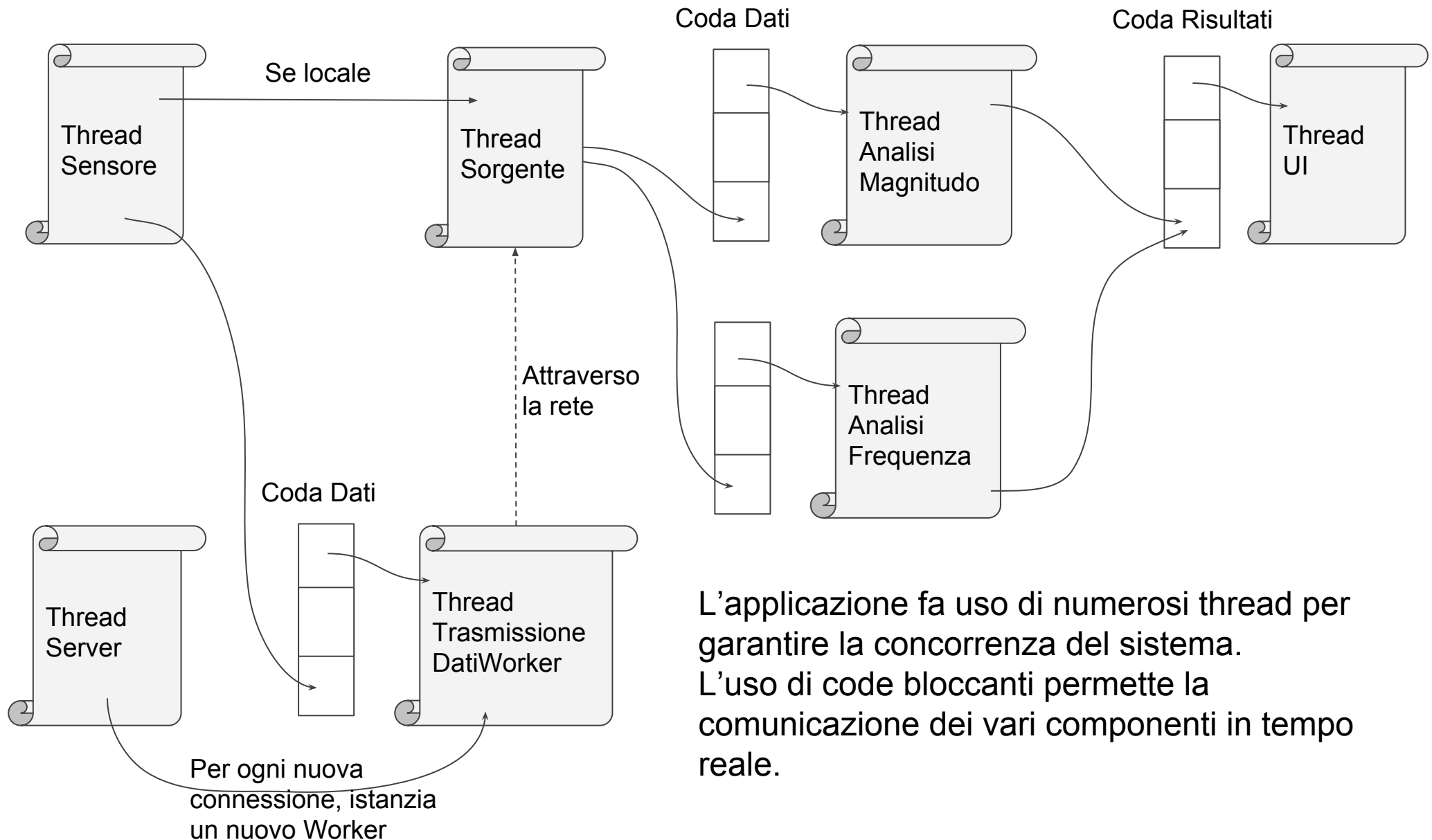
- **MaterialSkin**

Stile grafico secondo i principi di Google Material Design.
Vantaggioso per la creazione di nuovi componenti.
Scarsa possibilità di personalizzazione dell'aspetto grafico.

- **LiveCharts**

Visualizzatore di dati flessibile, semplice e potente.
Altamente performante e integrabile.
Notevole risparmio di tempo e carico di lavoro.

Architettura Multi-thread



Trasmissione remota



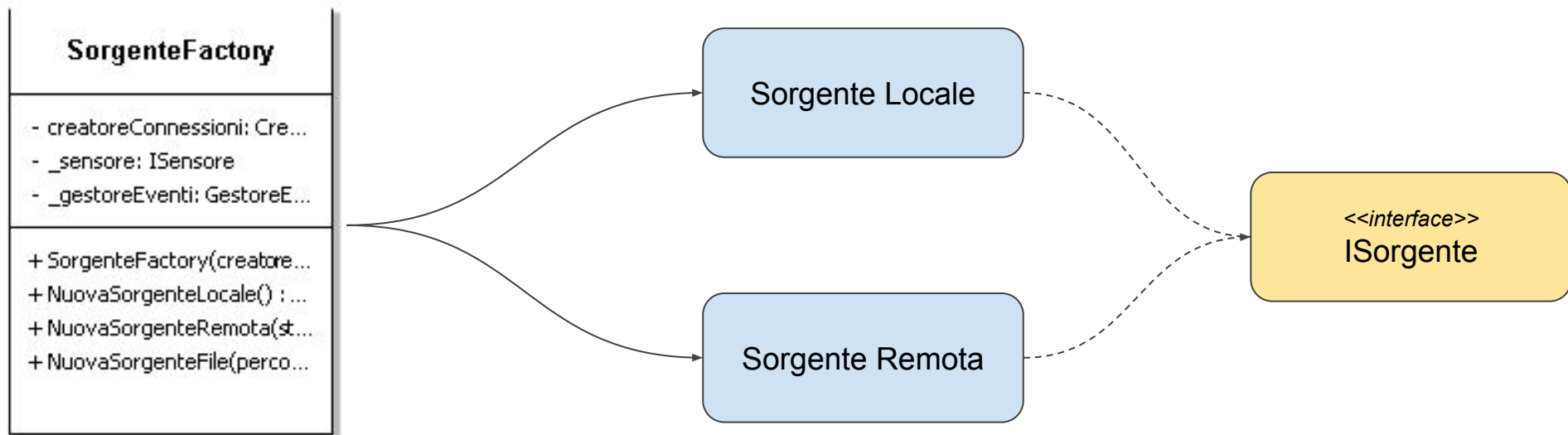
- Connessione **sicura** tramite protocollo **TLS**, implementato tramite la classe SslStream di C#.
- Utilizzo di **certificati autofirmati**, validati manualmente dall'amministratore (in maniera analoga al comando unix SSH).



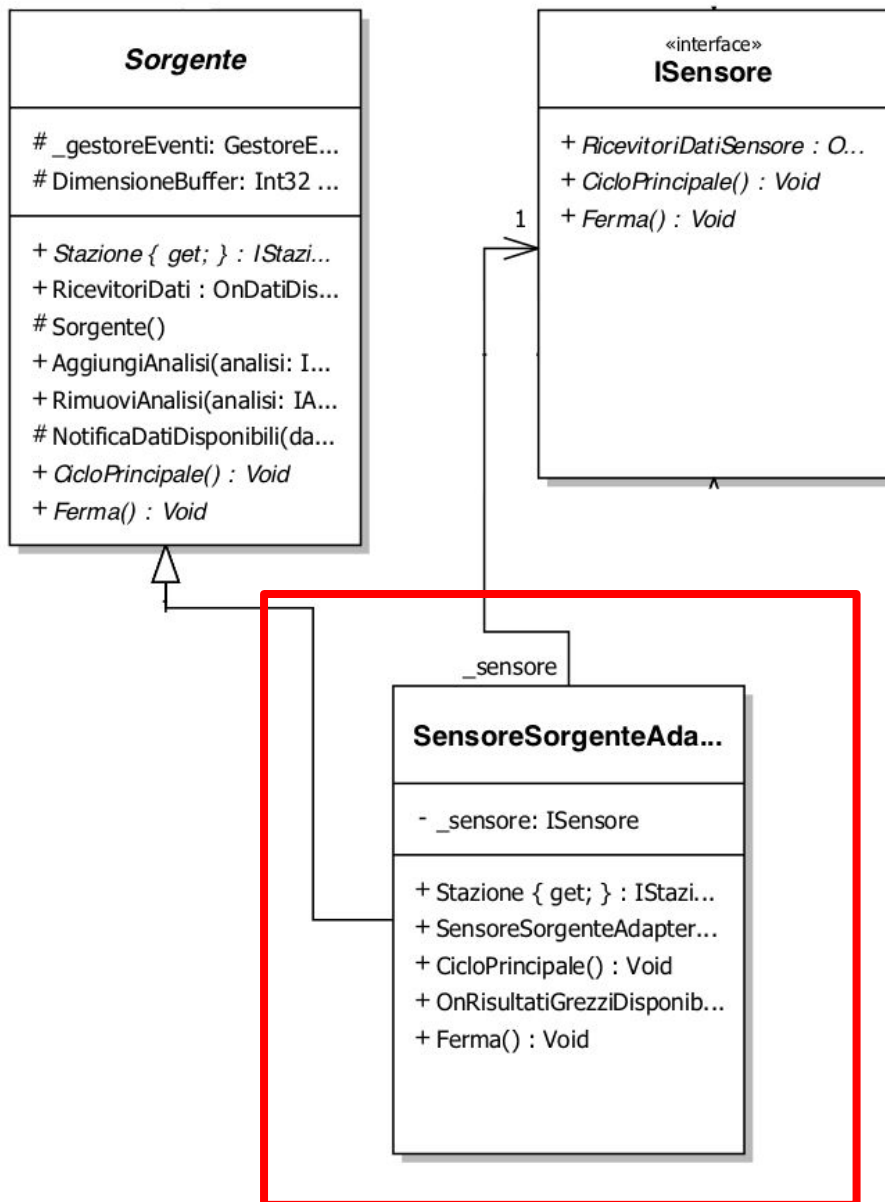
Astrazione della sorgente



- Il meccanismo delle sorgenti permette di nascondere al resto del sistema il fatto che il sensore sia locale o remoto.
- Per rendere trasparente la creazione di una sorgente, si utilizza il pattern **Factory**.

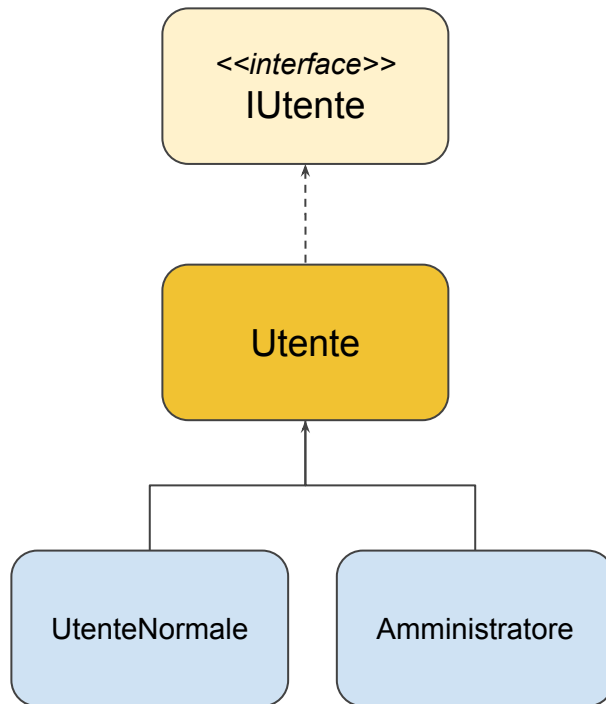


Pattern: Adapter



- Per mappare un sensore locale ad una sorgente, si utilizza il pattern **Adapter**.
- Questo permette di adattare l'interfaccia **ISensore** all'interfaccia **ISorgente**.

Uso della Reflection



L'istanza corretta della sottoclasse di Utente viene scelta a runtime in base al campo "type" salvato nella tabella.

- L'utilizzo della Reflection ha permesso di realizzare un sistema di mappatura tra dati memorizzati nel database e oggetti C# dinamico e **resistente a eventuali cambi futuri**.

```
public static Utente ConvertiRigaInUtente(SQLiteDataReader reader)
{
    // Ottengo il tipo dell'utente corrente
    Type tipoUtente = Type.GetType(typeof(Utente).Namespace + "." + reader["type"].ToString());
    if (tipoUtente == null)
        return null;

    // Genero un istanza per il tipo dell'utente corrente
    Utente utente = (Utente)Activator.CreateInstance(tipoUtente);

    // Popolo i campi dell'utente
    utente.Id = Convert.ToInt64(reader["id"]);
    utente.Nome = reader["nome"].ToString();
    utente.Cognome = reader["cognome"].ToString();
    utente.Username = reader["username"].ToString();
    utente.Email = reader["email"].ToString();
    utente.HashPass = reader["hashPass"].ToString();
    utente.SaltPass = reader["salt"].ToString();
    utente.LoginRemoto = (Convert.ToInt16(reader["loginRemoto"]) == 0 ? false : true);

    return utente;
}
```